University of California, Berkeley

Industrial Engineering & Operations Research

IEOR 290: Data Analytics and IoT: Machine Learning for Operations with Human Data Sources (Spring 2018)

Final Project: Prediction of Loan Default with Machine Learning Method

Author:
Wiseley Wu

May 11th, 2018

1 Introduction

Small loan is an important aspect of our everyday life: it allows aspiring entrepreneurs to get started on ideas that could be grown into business; it allows curious students to afford higher education that is otherwise unavailable without a stable income; more importantly, it allows ordinary people who have no friends or relatives for support to obtaining short-term financial assistance and get back on their feet to fight for the American Dream. Nevertheless, with loan it comes with the possibility of default as well. Default is a financial term describing the failure of meeting the legal obligation of a loan - paying back the principal and interest. It's a common problem in the financial industries and one of the major risks of offering loans. Of course, default does not happen the majority of the time and the lending banks usually able to make up the loss from a defaulting loan from other fully paid loans and their accompanied interests. Furthermore, banks issuing loans with higher interest rate to individuals with high probability of default - the financial institutions are trading off an increased chance of default with an increased profit from the high interest.

All things considered, default is a fact of life and most financial institutions have a well-established practice to minimize its impact and absorbing the loss. But what about a situation where instead of a single bank is issuing the loan, the loan is comprised of funds from several investors? LendingClub is one of the many peer-to-peer lending company that gives rise to this peculiar situation. In plain words, peer-to-peer lending company acts as a broker between borrowers and investors. The company creates a platform where borrowers can create small unsecured personal loans, and investors can seek out these loans and decide which loans to invest from. Borrowers obtain the loan they want, investors get to profit from the loan interest, and the company gets a cut from both parties (origination fee from borrowers and service fee from investors). This also means that when a loan goes default, it's no longer a single bank that is absorbing the loss - single or multiple individual investors will be absorbing it instead. The overall profit might be positive if all the loans were originated from a single lender as other fully paid loans could cover the loss, but this is no longer the case as there will be winners and losers among this new form of lending practices if the investors did not diversify.

An obvious solution to this problem is to predict whether a particular loan will go default based on initial information provided by the borrowers and their credit report. There's no doubt LendingClub already has an existing model in place to approve loans posted on their website. This paper will explore the process and result on formulating a new machine learning model that could predict a loan default; but more importantly, the model will focus on minimizing the overall loss in investment of bad loans in order to lessen the burden passed onto individual investors. As a side note, the paper will also explore privacy-preserving mechanism on sensitive information provided from the borrow's credit report. The end goal is to evaluate a simplified version of RAPPOR (Randomized Aggregatable Privacy-Preserving Ordinal Response) and determine whether data that have been hashed by this algorithm could still be use to predict loan default as stated previously.

2 Data

The dataset is provided by the peer-to-peer lending company LendingClub itself, which included all approved loans issued from the start of the company (2007) to the most recent quarter (2017 Q4). A membership was created as certain data are only accessible to LendingClub members [1]. Nathan George, a member on Kaggle.com, has already compiled all the loan data from 2007 to 2017 Q3 [2], so only the 2017 Q4 data (in CSV format) were needed to append to the existing dataframe. All in all, there is a total of 1765426 approved loan instances and 152 features ranging from a borrower's debt-to-income ratio (DTI), FICO score, earliest credit line, and more. Due to the nature of the problem stated in the introduction, features that are only available after a loan has been approved (loan interest rate, installment plan, LendingClub's assigned loan grade, payment received to date, etc.) will not be used in the predictive model as those are information otherwise not readily available to the company at the loan approval stage. [Figure 1] showed that more than half the loan data have "current" status - active loans that are neither fully paid nor default. Those loan instances were not considered and were removed from the data.

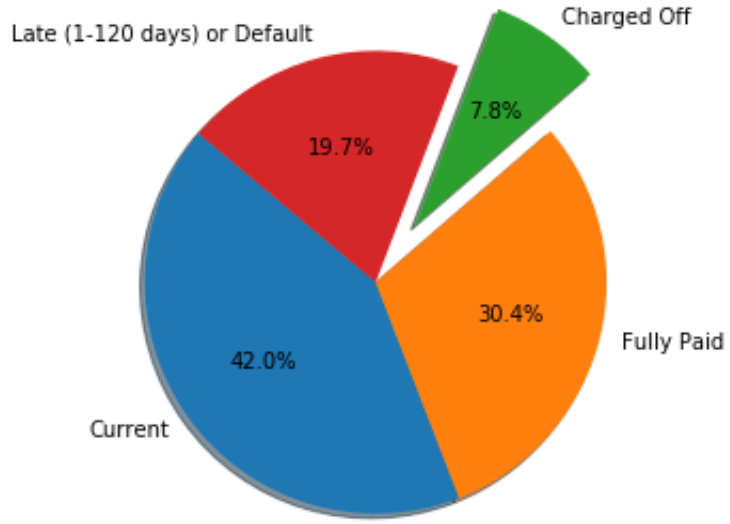

Figure 1: Breakdown of Loan Status

Since both "Late" and "default" indicated the loans were still in service and had not been terminated, those instances were also removed along with the "current" status loans. By removing all loan instances except "Fully Paid" and "Charged Off", the problem had been modified to a binary classification problem on whether a given loan will be defaulted (Charged Off) or not. The term default and charged off would be used interchangeably throughout the paper, but they both have the same meaning where there is no longer reasonable expectation for the borrower to pay off the loan and prevent a charge off. The end result is a data set

with 820554 observations, with 79.5 % of them being fully paid loans.

It is noteworthy to point out a potential sampling bias in this dataset. Since the loan data were provided by LendingClub as is, it's at their discretion of whether releasing all the default data to the public or not. It's entirely possible that some default data were hidden from the public view in order to inflate the number of non-default loans and ease investors' confidence. Additionally, the data only included loans LendingClub had approved, which meant some form of prediction had already been done to evaluate each loan's default risk. A separate model created on data filtered by the initial model would certainly introduce further sampling bias in this dataset.

With some disqualifying features removed, there were still over 80 features that could be used for the predictive models. However, a closer inspection of these features revealed many missing (NaN) values, with some of them contained more than hundred thousand rows of NaN values. Many of these features contained data with a value of 0, which eliminated one of the options of imputing these missing values with 0 as they might possess different meaning from the real 0s. With no reasonable expectation to impute these missing data, a decision was made to remove any features that have over 100,000 row of NaN values, and the remaining NaN values were removed via the removal of the entire loan instance (row) itself. This process reduced the dimension of the dataframe from 820554 rows and 87 features to 661867 rows and 67 features, a sizable reduction. A quick check on the dependable variable (loan status) showed that the percentage of fully paid loan still took up 79.7 % of all data to ensure the missing data removal did not remove one side of the loan status discriminatively.

Additional rows of data were also removed. For example, one row of data was removed due to it having a negative value of debt-to-income ratio (which should only be a positive value). Another row of data with a loan purpose of "educational" was removed since it was the only instance with that particular purpose. The same concept was also used to remove 167 rows of data due to their niche response of their home ownership status (any, other, or none instead of mortgage, rent, or own). On the other hand, missing values from *employment title* were imputed with string "None" to differentiate them from the rest of loan instances which provided proper title names.

With the data preprocessing completed, some additional features were implemented using the loan issuing year and month as a starting point. First of all, the 3-month LIBOR rate (in USD) was added to keep track of the short-term loans rate at the time of loan issue [3]. LIBOR, or Intercontinental Exchange London Interbank Offered Rate, is the world most widely used benchmark for short-term interest rates. Its primary purpose is to serve as a benchmark for debt instruments such as mortgages, student loans, and credit card rates [4]. There's no doubt the rates offered by LendingClub are highly influenced by LIBOR, and the rate itself could potentially impact whether a borrower could pay off the debt or not. Secondly, the unemployment rate of United States was incorporated to the data set [5]. This

is a major benchmark used by economists to monitor the health of the United State's job market, thought it has its fair share of weaknesses as a low employment rate could result from a shrinking workforce instead. Nevertheless, this number provided a good indication of the percentage of American workforce who are out of a job and might rely on loans to support their everyday needs. Finally, the S&P 500 stock index at the time of issuing the loans was incorporated to the data as well [6]. S&P 500, or Standard & Poor's 500, is an American stock market index based on market capitalizations of a diverse mix of 500 U.S. companies. It's one of the most followed indices, often considered the best representations of the U.S. stock market and bellwether for the U.S. economy [7]. This was added as another indication of economy along with unemployment rate to see whether a particular loan was issued during a tremorous economy, which could increase the probability of default.

A pairwise correlation plot [8] was produced to observe whether some features were closely related and correlated with each other [Figure 2]. There were several strong positive and negative correlations between features. Close inspection of these features revealed their similarity in definitions (number of satisfactory accounts vs. number of open credit lines, number of bankcard accounts vs. number of satisfactory bankcard accounts, etc.). Some of these features were removed in order to reduce multicollinearity in the predictive models.

The last step in data processing involved the use of one-hot-encoding to convert all categorical features to their respective binary columns of 1s and 0s. On the other hand, continuous features were standardized to a distribution with zero mean and a unit variance. The dependent feature, loan status, was also encoded such that the value of 1 represents a default loan and 0 represents a fully paid loan. Finally, the data were randomly split into 3 parts: training data set which contained 595370 (90 %) loan instances, validation data set which contained 33076 (5 %) loan instances, and test data set which contained 33077 (5 %) loan instances. The reason data were not split temporally (use old data for training and new data for validation and testing) was due to the fact that it might further introduce sampling bias. As the analysis approached to recently approved loan data, less default loans would be found as the data were filtered by loans that were either completed (fully paid) or not (default), and recently approved loan data were more likely to be fully paid since it takes a certain period time to confirm whether a borrower truly decides not to continue the payment (from the earlier definition of default, as short as 1 month and as long as 5 months). Therefore splitting data by date might introduce bias to the dataset, and random splitting was conducted instead.

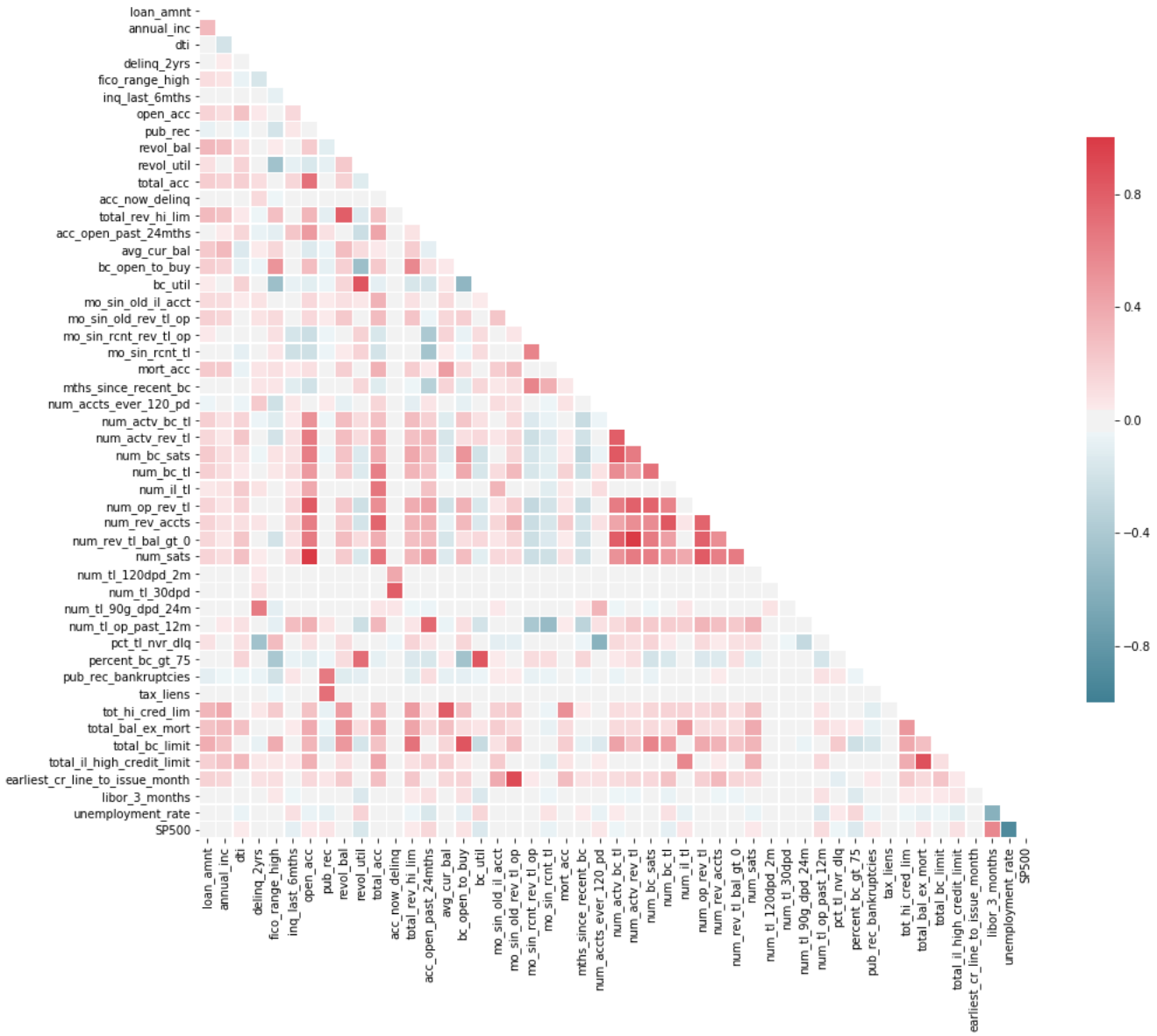

Figure 2: Diagonal Correlation Matrix between Features

3 Model

The goal of this paper is to engineer a predictive model that could discern a good loan from a bad loan (likely to default) with data available only before and during the loan approval process. As a binary classification problem the most simple metric is the measurement of accuracy, or the percentage of correct loan status assignment out of all possible loan instances. Nevertheless, due to the imbalanced nature of the data where 79 % of approved loans ended up fully paid, a simple baseline model where predicting all loans to be fully paid will easily achieve an impressive looking 79 % accuracy, but with absolutely no meaning behind it other than blind guess. Furthermore, without adjustment to the input, some models presented with a highly imbalanced training data might actually came to the same conclusion as the baseline model: simply predicting the majority loan status. Therefore cares must be taken when setting up the input data and identifying a proper metrics to evaluate various predictive models.

Metrics

Since one of the main goals of the model is to reduce the overall investment loss, one way to formulate the problem is maximizing return of investment (ROI), which could be defined as:

$$\text{ROI} = \frac{\text{gain from investment}}{\text{cost of investment}} - 1$$

However, one unintended consequence of defining the optimization problem as maximizing ROI could be the model learning some loan instances having higher weights than others (higher loan value), resulting in a model that prioritizes high value loan over low value loan. Instead of maximizing return of investment, it would be better to train a general model that could increase the probability of assigning the correct status label to a loan instance correctly. This could be succinctly captured by the ROC AUC score, or the area under the curve of the receiver operating characteristics. Another metrics that could capture the imbalanced nature of data is the Cohen's kappa score, which formally defined as:

$$\kappa = \frac{\text{observed accuracy} - \text{expected accuracy}}{1 - \text{expected accuracy}}$$

In essence, kappa score is a measurement of how similar the predicted labels from the truth, normalized by the performance of a naive classifier which always predict the high frequency value [9]. In conclusion, the predictive model was trained to maximize ROC AUC score, but other metrics such as kappa score and ROI were monitored to compare models trained with different methods.

Furthermore, two sampling techniques were implemented during the course of various model trainings. As model hyperparameters were being tuned, the majority class (fully paid loans) was under-sampled such that the re-sampled training data set was well-balanced between the two classes. Once tuning was completed, the final prediction model was trained using a bootstrapped training set where the minority class (default loans) was over-sampled (with replacement) to create a balanced data set. These methods were implemented to ensure the model would not produce an unsatisfactory result of predicting same class for all loan instances.

Other metrics used for models evaluation included: *current ROI total*, which represents the current cumulative return of investment using LendingClub's strategy; *model ROI total*, which represents the return of investment if the model in question were to be adapted in picking which loan to be approved by LendingClub (a prediction of default loan would naturally leads to its rejection); *fraction approved total*, which simply shows the fraction of loans that were approved by the model.

Baseline Model

As mentioned in the previous section, baseline model was simply a naive model of predicting the most probable outcome for all loan instances. Since the data has more fully paid loan than default loan, a baseline model will predict fully paid loan for all instances.

Logistic Regression

Logistics Regression is a simple classification method which seeks to predict the probability of a binary outcome by modeling the dependent feature as a Bernoulli random variable. The conditional probability of a positive outcome (1 instead of 0) could be generalized with the formula below:

$$P(Y = 1|X) = \frac{1}{1 + e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\cdots+\beta_p X_p}}$$

Where Y=1 represented a given loan instance would end in default, X_i is the i-th feature data, β_i is the coefficients for the i-th feature, and p is the number of features (69 with this data set). The coefficients are determined using maximum likelihood estimation (MLE), which maximize the probability of observing a given data set. The advantages of logistic regression lies in its simplicity in implementation and the ease of interpretation, as each coefficient describes the magnitude and direction each feature has in determining the classification of each observation. Its simplicity nature meant the model could be trained and tuned very efficiently.

The other side of the coin for a simple model like logistics regression is the difficulty for it to model complex, non-linear behaviors. As seen in the equation above, the result is dependent on the weighted sum of the features and their coefficients. This created a great generalized linear model with low variance, but its high bias might make it ignorant of some non-linear relationship between the features and labels.

Support Vector Machine

Support Vector Machine (SVM) is a supervised learning method with the goal of constructing a hyper-plane in a high-dimensional space, which could be used to segregate different populations. A good support vector machine needs to create a hyperplane (or multiple hyperplanes for multi-classification) which maximize the distance to the nearest training data points of any class (margin support vectors) as this would lower the generalization error of the classifier [10]. This could be expressed with the following optimization problem:

$$\min_{\beta,\beta_0} \frac{1}{2}||\beta||_2^2 + C\sum_{i=1}^{n}\zeta_i$$
$$\text{subject to } y_i(x_i^T\beta + \beta_0) \geq 1 - \zeta_i$$

Where the problem's goal is to minimize the norm in addition to a user assigned penalty weight C while subjected to the support vector constraint.

Due to the nature of support vector machine, all data points other than the margin support vectors become redundant once the optimal hyperplane is found. This means small change to the data would not affect optimality of the hyperplane, resulting in a model that generalizes very well and prevents overfitting. An additional advantage of support vector machine is the ability to define a kernel which projects the data into a space where they are linearly separable. In this paper, linear and RBF kernel were applied to the model training. The linear model provided by *scikit-learn* is another implementation of SVC (support vector classification) called LinearSVC, which is highly scalable to large data. Additionally, it provides coefficients in its output which allows level of interpretation that's similar to logistics regression, making it a strong, generalized predictive model with high interpretability. Of course, similar to logistics regression, it falls short on complex and nonlinear behavior, which could be handled with RBF (radial basis function) kernel. The kernel applies an exponential transformation to the squared Euclidean distance between two feature vectors, allowing it to model nonlinear behavior. In exchange, the model loses the interpretability of the linear version; it also requires much longer computational time to find the optimal hyperplane.

Gradient Boosted Trees

Gradient Boosted Trees is a variant of existing tree-based method such as CART (Classification and Regression Tree) and Random Forest (an ensemble combination of deep CART). A simple decision tree works by making numerous split at different features in order to classify individual data points. A CART is similar to a decision tree, but instead of the leaf containing the data points, each leaf is assigned a score for the purpose of optimization. A model built with gradient boosted trees is essentially comprised of a series of weak (shallow) trees; the model combines these trees adaptively where the current model performs well on some of the data that the previous model did not perform well on [11]. The iterative learning process could be represented by the following equations:

$$\hat{y}_i^{(0)} = 0$$
$$\hat{y}_i^{(1)} = f_1(x_i) = \hat{y}_i^{(0)} + f_1(x_i)$$
$$\hat{y}_i^{(2)} = f_1(x_i) + f_2(x_i) = \hat{y}_i^{(1)} + f_2(x_i)$$
$$\dots$$
$$\hat{y}_i^{(t)} = \sum_{k=1}^{t} f_k(x_i) = \hat{y}_i^{(t-1)} + f_t(x_i)$$

Where t represents the total number of iterations. This creates a very powerful learning method which adapt the weakness as the tree is boosted in each iteration, but this also attributes to one of its weaknesses: the fact that tress are boosted iteratively means the process could not be sped up by parallelizing the training process itself. Hence, this method is not as efficient as the closely related random forest method. Additionally, the model provides limited interpretability like other tree models due to its complex ensemble nature. Nevertheless, the model outputs "features importance", which weights the importance of each feature by measuring the metrics improvement as the tree splits with that particular features. Therefore, a high importance value of a feature indicates its necessity in making the classification itself. This does not provide the same level of interpretation as logistics regression, where one could make inference with the signs and magnitudes of each feature's coefficient, but the information is helpful regardless. Of course, with the loss of interpretability, the model excels in modeling nonlinear behaviors due to the nature of decision trees.

In this paper, XGBoost, a variant of Gradient Boosted Trees was used to train one of the predictive models. It is highly memory efficient, which enables it to scale to large data set. It is also well known in many data contests and frequently used by top scorers in Kaggle competitions.

Wide and Deep Neural Network

The last model evaluated in this paper is a combination of wide and deep neural network [12] (henceforth referred as wide-n-deep model) as implemented by Google's *TensorFlow*. What made this model fitting for the LendingClub data is its ability to memorize interactions between features, while generalizing the model via the use of deep neural network to aid the exploration of new features combinations that have rare or no occurrence in the past. The dataset in question has many categorial features, including zip code, state, job title description, and purpose of loans that contained more than thousand unique values; one-hot-encoding these features would translate them into high dimensional, sparse binary columns that add enormous burden to most conventional machine learning model trainings. However, *Tensorflow* could transform these high-dimensional sparse columns into low-dimensional dense real-valued vectors called embedding vectors, which could then be fed into the neural network. These dense vectors allow the neural network to learn the overall patterns of location and distance between vectors that are otherwise impossible to discover by one-hot-encode binary vectors.

As the name suggested, wide-n-deep model is an implementation of machine learning model with a wide component and deep component. The wide component consists of a generalized linear model of $y = w^T x + b$, where y represents the label, x represents features, w represents feature coefficients, and b represents the bias. In order to implement memorization of sparse interactions between features, cross-product transformation could be used between two binary columns to capture their interactions. The transformed value could only be 1 if both the values used in the cross-product are 1, otherwise it's 0. This also introduces non-linearity to the linear model which could capture complex behavior within the data. With the LendingClub dataset, features such as *employment length*, *homeownership status*, *verification status*, *loan purpose*, *address zip code*, and *address state* were used in the wide component. Additionally, cross-product transformations of *employment length* and *homeownership status* / *loan purpose* were implemented to capture any sparse interactions between these features.

On the other hand, the deep component is a feed-forward neural network where dense value vectors could be fed into the hidden layers in order to generate new feature combinations for predictions. In addition to the continuous columns and embedding columns of *address zip code* and *address state*, a text embedding column trained on English Google News 200B corpus was also included to handle the employment title provided by the borrowers. These low-dimensional, dense real-valued vectors in addition to the continuous vectors were then fed into the hidden layers of the neural network in the forward pass. Each hidden layer performed the following computation:

$$a^{l+1} = f(W^l a^l + b^l)$$

where l is the hidden layer and f is the activation function. a^l, b^l, and W^l represents the activation, bias, and model weights at the l-th layer. The wide and deep component were

combined using a weighted sum of their output log odds as the prediction, and was fed to a logistic function for joint training, and a sigmoid function was used to bound the result between 0 and 1.

Even though the dataset benefited a lot from using a wide-n-deep model to model categorical features with many possible values, the complexity of setting up a deep neural network model resulted in significant amount of work compared to most conventional machine learning model. Additionally, the highly non-linear nature of neural network model meant the local minima found by the algorithm might be extremely volatile and might behave badly on unseen data that are vastly different from the training and validation data. These difficulties should be taken into account when evaluate this model with others.

The wide-n-deep model was setup with the following parameters. Follow-the-regularized-leader (FTRL) algorithm with L_1 regularizer was used as the linear optimizer. Adagrad was used as the deep neural network optimizer. The concatenated dense vectors were fed into 2 rectified linear units (ReLU), with 100 and 50 hidden units respectively. For each epoch, the dataset were rebalanced [13], shuffle, and sampled with a batch-size of 5000 for model training, and the entire validation data set were used for model evaluation. After the model was trained fro 25 epochs, the final model was evaluated again with the full set of validation data.

4 Result and Analysis

The baseline model (naive model that predicts the most probable outcome for all loan instances) was used to set the baseline performance where other predictive models should at least able to achieve, if not surpassing it. The baseline model also represented current strategy used by LendingClub to approve loans.

All models summary metrics could be found in [Table 1]. Baseline model had the *Model ROI Total* value at -0.117, which meant using baseline model resulted in a negative return of investment if all the loans were invested by a single entity. This also represented the *Current ROI Total* which was used as a baseline comparison with other models. The model also scored a 1 on *Fraction Approved Total* since all loans were approved. Other baseline metrics were as expected: *Cohen Kappa* and *Default Recall Score* were both 0 since the model only predicted loans to be fully paid, and *ROC AUC* Score was 0.5 as expected of a baseline model. Accuracy was however at close to 80 % since majority of loans were fully paid, it made sense a naive model of predicting all loans were fully paid was bound to score highly in accuracy - which is why it's more important to focus on metrics such as *Kappa*, *ROC AUC*, *Recall*, and *ROI* instead.

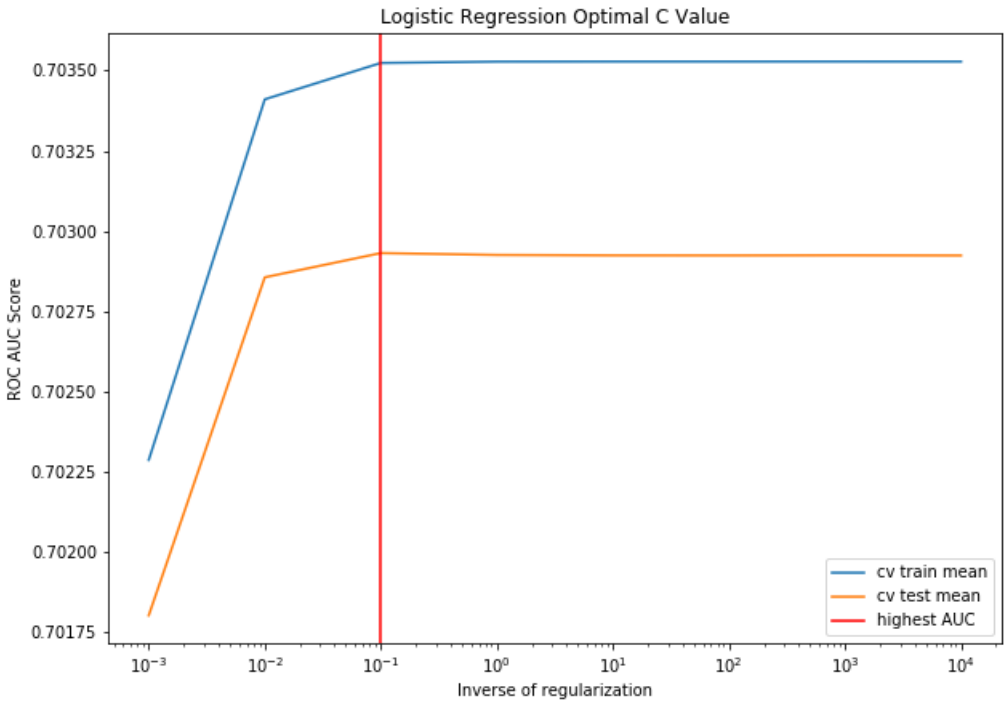

Figure 3: ROC AUC value vs different values of C used in logistic regression model training

Logistic Regression

A parameter search was conducted on the logistic regression model to identify the value of inverse of regularization providing the highest *ROC AUC* value. Inverse of regularization, or *C*, trades off misclassification of training data against simplicity of the model. A small value specifies stronger regularization. The under-sampled training data were split into 5 folds and cross-validated with each other set. [Figure 3] showed the result of the search, and 0.1 was chosen for the final model training with training data. Selected feature coefficients of the trained model were shown in [Figure 4]. Feature *purpose_small_business* having the largest positive coefficient suggested that a loan that is for small business is very likely to go default. On the other hand, *total_acc* having the largest negative coefficient suggested the more accounts (bank, credit card, loan, etc.) a borrower has, the less likely his or her loan will go default. Other intuitive observations including: high debt-to-income ratio *dti*, loan amount *loan_amnt*, and account balance excluding mortgage *total_bal_ex_mort* contribute to default, while a high installment credit limit *total_il_high_credit_limit* contribute to a fully paid loan as high installment credit limit is only obtainable if borrowers paid their bills on time and were generally in good financial standing. Interestingly, *unemployment_rate* having a negative coefficient suggested that a loan issued during high unemployment rate will less likely to go default, which was unexpected.

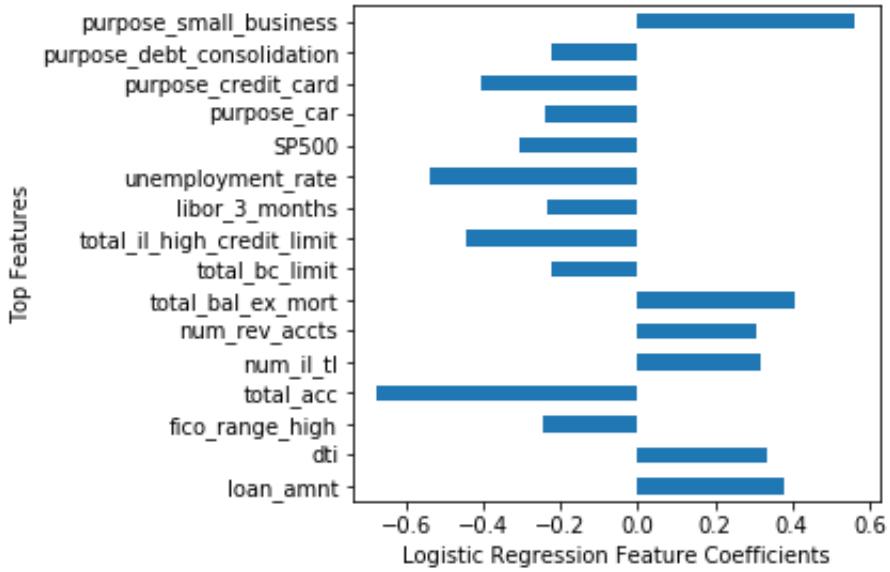

Figure 4: Top features' coefficients as observed by the logistics regression model

[Table 1] summarized the metrics of the logistic regression model on the validation data set. The ROI showed a healthy improvement from -0.117 to -0.054, suggesting investors would have lost less money overall if the loans were approved based on the logistic regression model.

In return, 43 % of loans that were otherwise approved by LendingClub would have been rejected by this model. Both ROC AUC score and Cohen Kappa score expectedly showed significant improvement (0.70 and 0.21 respectively) compared to the baseline model. A default recall score of 0.66 suggested that the model was able to discern 66 % of all default loans. The accuracy of the model was 63.7 %, a drop from baseline model's 79 %. But this could be a decent compromise to minimize investors' overall loss.

Support Vector Machine with Linear Kernel

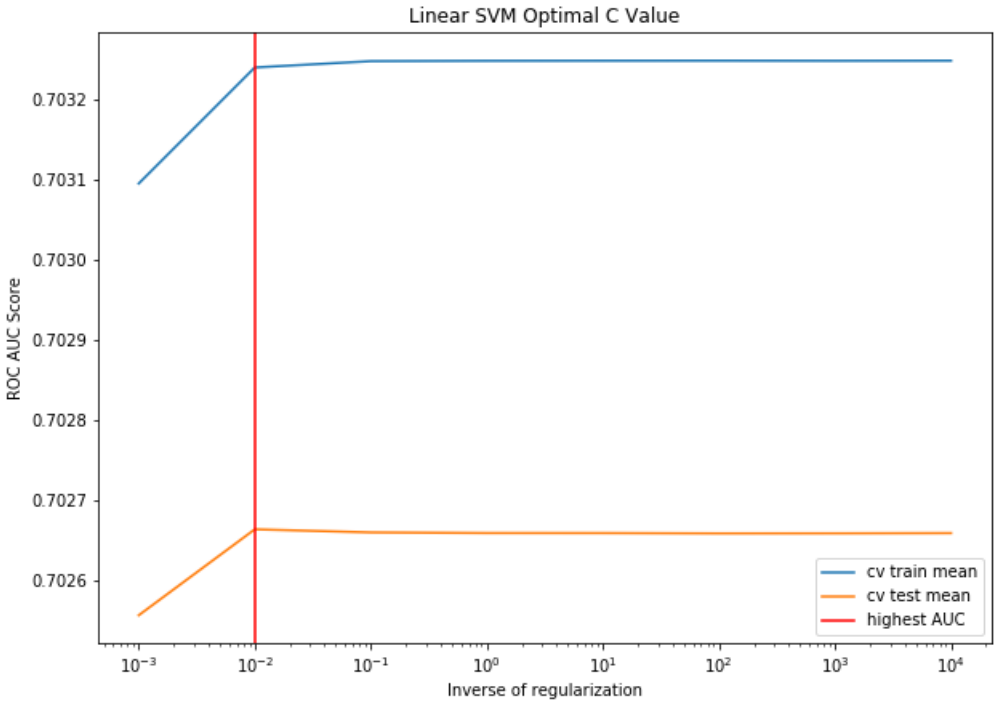

Figure 5: ROC AUC value vs different values of C used in linear SVM model training

A similar parameter search was also conducted on the linear SVM with the same range of inverse of regularization used by logistic regression, where a smaller C results in a smoother decision surface and a larger C results in a flexible model that strives to classify all training data correctly. The under-sampled training data were split into 5 folds and cross-validated with each other set. [Figure 5] showed the result of the search, and the value 0.01 for inverse of regularization resulted in a model with the highest AUC. The chosen value was used to train the final model, and selected feature coefficients of the trained model were shown in [Figure 6] - it's apparent linear SVM model and logistic regression model shared many of the features with similar large negative and positive coefficients.

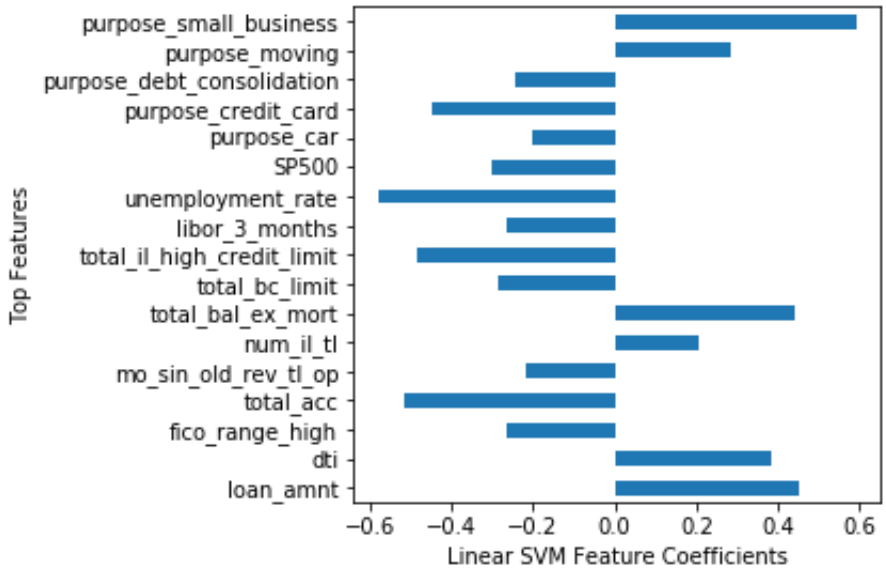

Figure 6: Top feature coefficients as observed by the linear SVM model

Linear SVM model had a slightly lower accuracy compared to logistic regression at 63.0 %, but did a slightly better job at discerning default loans, scoring 0.67 at default recall. AUC and ROI of both models were very similar, though linear SVM model rejected slightly more loans, with an approval fraction at 0.561.

Support Vector Machine with RBF Kernel

In addition to inverse of regularization, SVM with RBF kernel allows the fine-tuning of *gamma*, which defines how much influence a single training data has. The larger the *gamma*, the closer other data have to be in order to be affected. Due to the significant time required to train a SVM model with RBF kernel (non-linear), the rebalanced under-sampled data were further subsampled to 10 % before using them for grid search. Once again, The subsampled training data were split into 5 folds and cross-validated with each other set. [Figure 7] showed the resulting grid search in a heatmap, where an inverse of regularization of 1.0 and gamma of 0.01 produced a model with the highest AUC. Gamma beyond 1.0 had significant reduced performance, as an increased value of gamma required all data points to be relatively close to each other - which might not necessary be the case. The final SVM model with RBF kernel was trained using the optimal parameter; since the kernel was non-linear, feature coefficients were unavailable and no inference could be made with this model.

Accuracy of the final SVM model with RBF kernel was comparable to both the linear SVM and logistic regression model, though it had a higher default recall of 0.71, suggesting it could discern 71 % of all default loans [Table 1]. This resulted in a much improved AUC at 0.721, and Kappa at 0.233 compared to the previous two models. The return of investment

also improved slightly at -0.043, though in exchange more loans were rejected compared to linear SVM and logistic regression models. [Table 1]

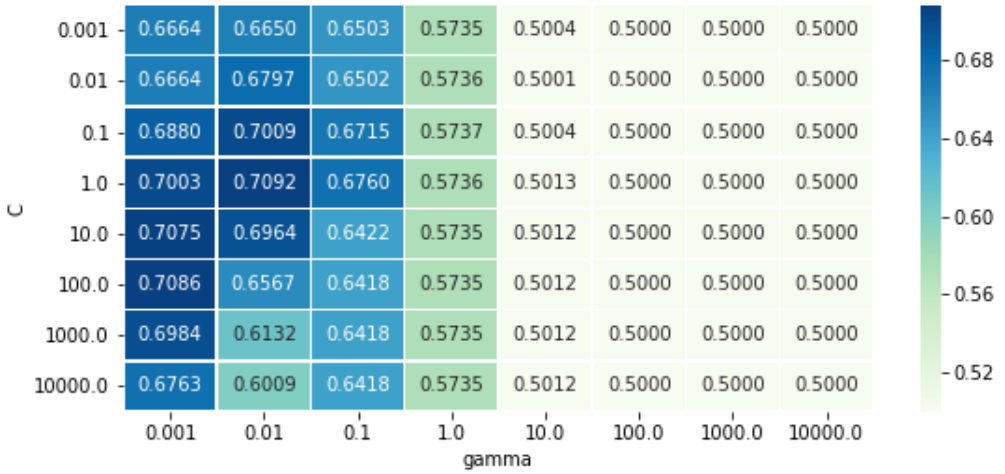

Figure 7: Grid search of C and *gamma* to maximize AUC value (shown inside the cell) of the SVM model with RBF kernel

Gradient Boosted Trees (XGBoost)

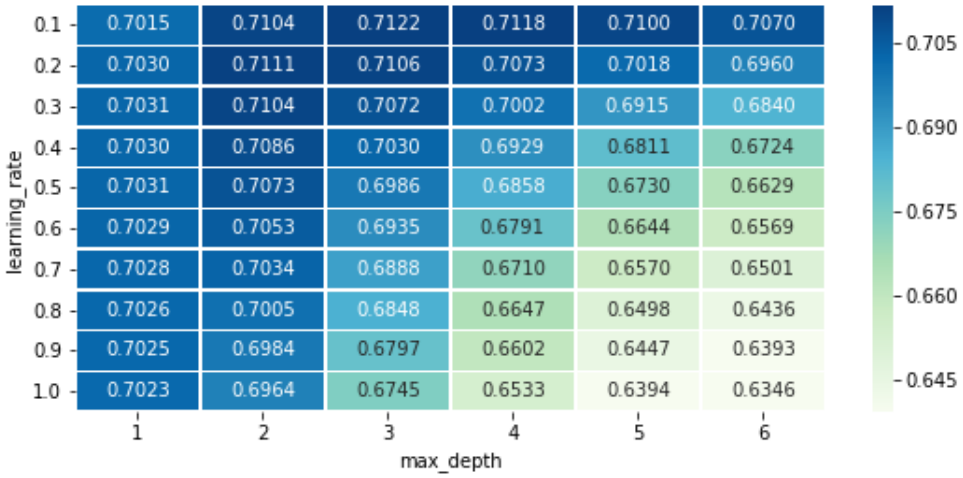

Figure 8: Grid search of *max depth* and *learning rate* for XGBoost model. Value of 3 and 0.1 created a model with the highest AUC (value shown in the cell)

Using the under-sampled training data, three grid searches were conducted to identify opti-

mal hyperparameters that maximized ROC AUC score [14]. The first grid search identified parameters that were later used in the subsequent grid searches - each of them was built on parameters identified by previous searches. These parameters include: *learning rate* - which controls the shrinkage of feature weights, *max depth* - the maximum depth of a tree, *min child weight* - the minimum sum of instance weight of a leaf before giving up partitioning, *colsample by tree* - the fraction of features to be used when constructing each tree, and *subsample* - the fraction of data randomly chosen when growing a new tree. [Figure 8] showed the first grid search between *learning rate* and *max depth*, and the values of 0.1 and 3 respectively were identified to be optimal. A further fine tuning for a smaller learning rate was conducted along with *min child weight* as shown in [Figure 9], and 0.09 and 7 respectively were identified to be optimal. It's also interesting to see that *min child weight* did not impact the model's AUC all that much. Finally, the third grid search was conducted between *colsample by tree* and *subsample*, where 0.9 and 0.8 respectively were identified to be the condition resulting in highest AUC [Figure 10].

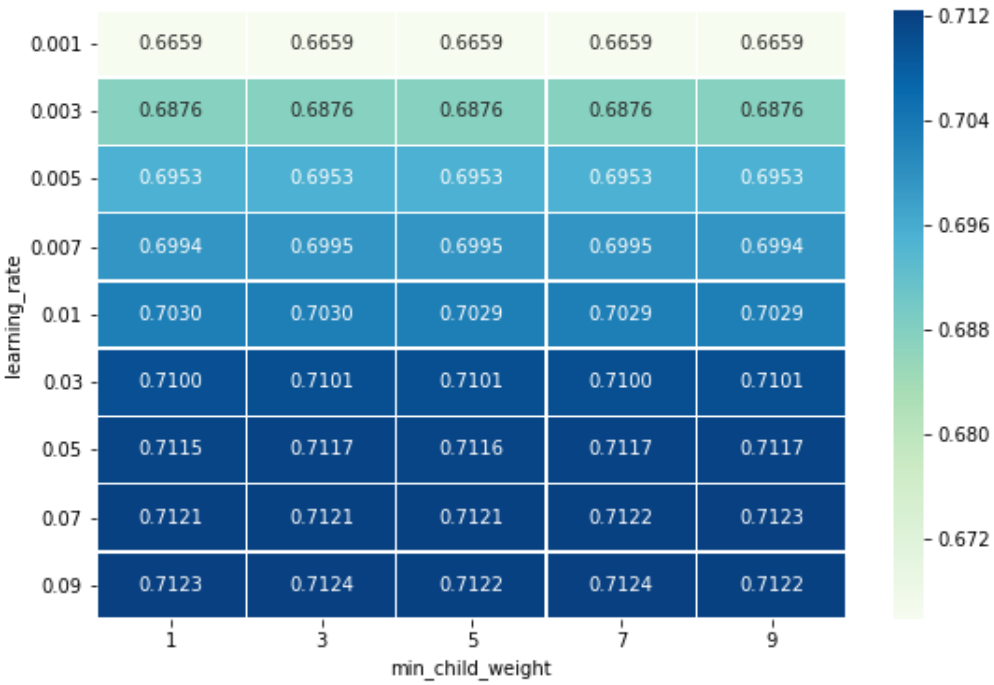

Figure 9: Grid search of *min child weight* and *learning rate* for XGBoost model. Value of 7 and 0.09 created a model with the highest AUC (value shown in the cell)

With the optimal parameters in place, the final gradient boosted trees model was trained with those optimal parameters, along with a maximum of 10000 boosting trees. The model was evaluated every round (tree) using the training and validation set, and model training

was terminated if the AUC of validation set were to not improve for 500 rounds. [Figure 11] showed the resulted model training, where the process terminated at 1464 trees with an AUC of 0.722. It's unsurprisingly to see the training set AUC continued to improve while validation set AUC did not, a sign that using a model with more boosted trees might potentially overfit the data and result in poor result with test set.

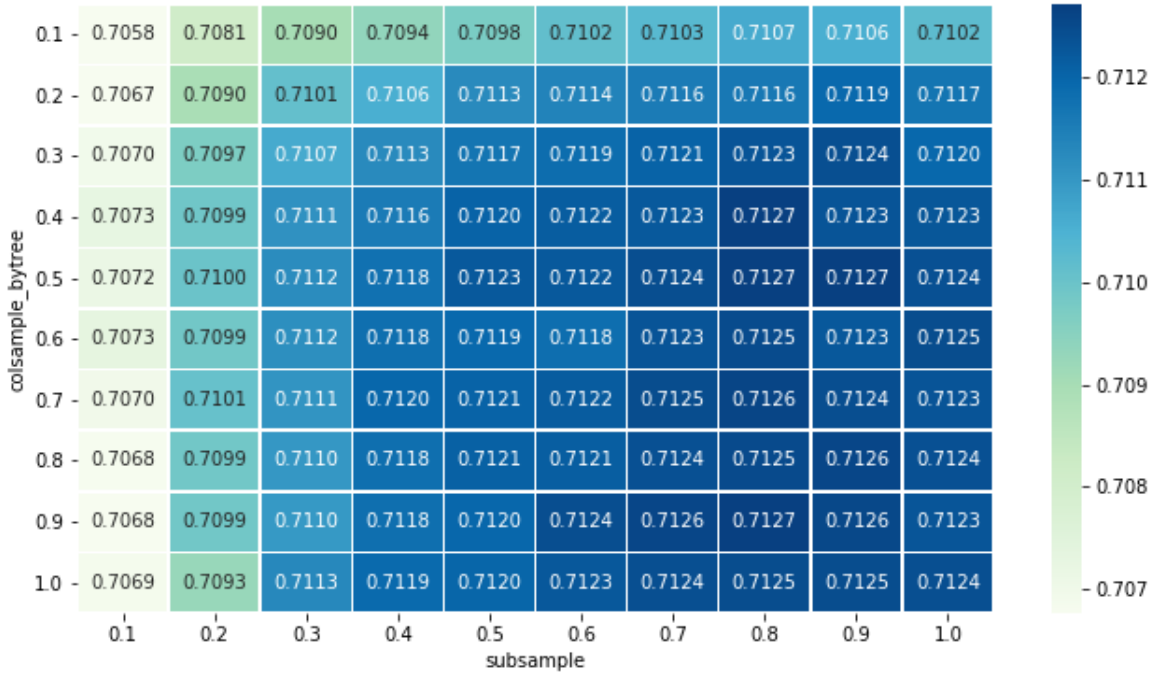

Figure 10: Grid search of *subsample* and *colsample bytree* for XGBoost model. Value of 0.8 and 0.9 created a model with the highest AUC (value shown in the cell)

Similar to SVM with RBF kernel, gradient boosted tree is a non-linear modeling algorithm and it does not provide simple interpretation and inference compared to linear regression or linear SVM model. Nevertheless, gradient boosted tree provides features importance as mentioned in the previous section, and a high importance value indicates its necessity in forming the classification model. [Figure 12] showed some of the top relative importances from the model, where annual income, loan amount, debt-to-income ratio, total balance excluding mortgage, and total installment credit limit contributed the most in AUC improvement when the boosted trees were splitting with these features. Some of these features were also seen in previous linear models with high coefficient values.

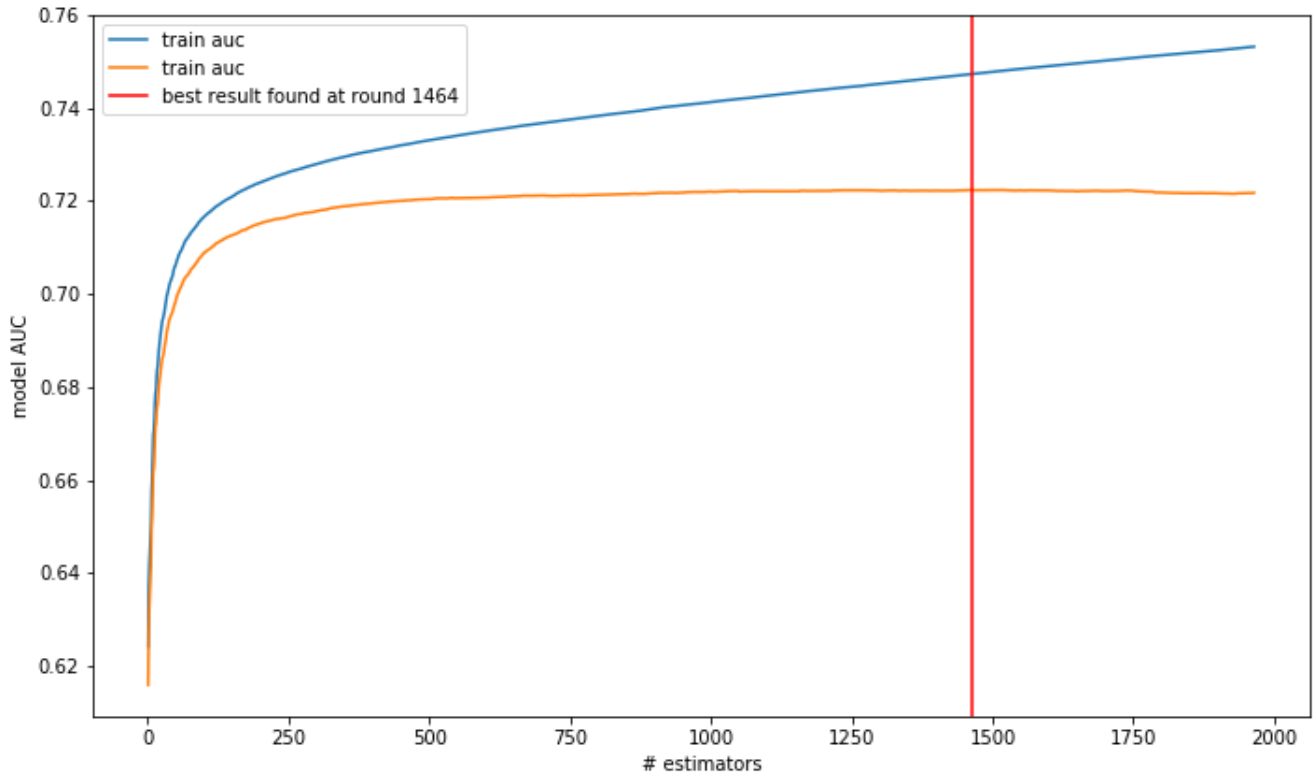

Figure 11: AUC vs number of estimators in the final XGBoost model. The AUC from validation data set did not improve further after 1464 trees were used

The final model scored an accuracy of 64.3 % and a Kappa of 0.226 on the validation model, outperforming all the previously trained models. This was achieved by balancing the classification of default and fully paid loans without putting too much weight on one or the other, evident by the default recall score of 0.68 and fully paid recall score of 0.63. The AUC, however, was similar to the value obtained by the SVM model with RBF kernel. Even though the return of investment of -0.046 did not outperform the same kernel, the fraction approved was much in line with the logistic regression model while retaining the ability to discern more default loans [Table 1]. This stuck a great balance in identifying default loans without rejecting overwhelming number of loans that were otherwise fully paid - as doing so will result in the overall revenue drop even though return of investment would increase.

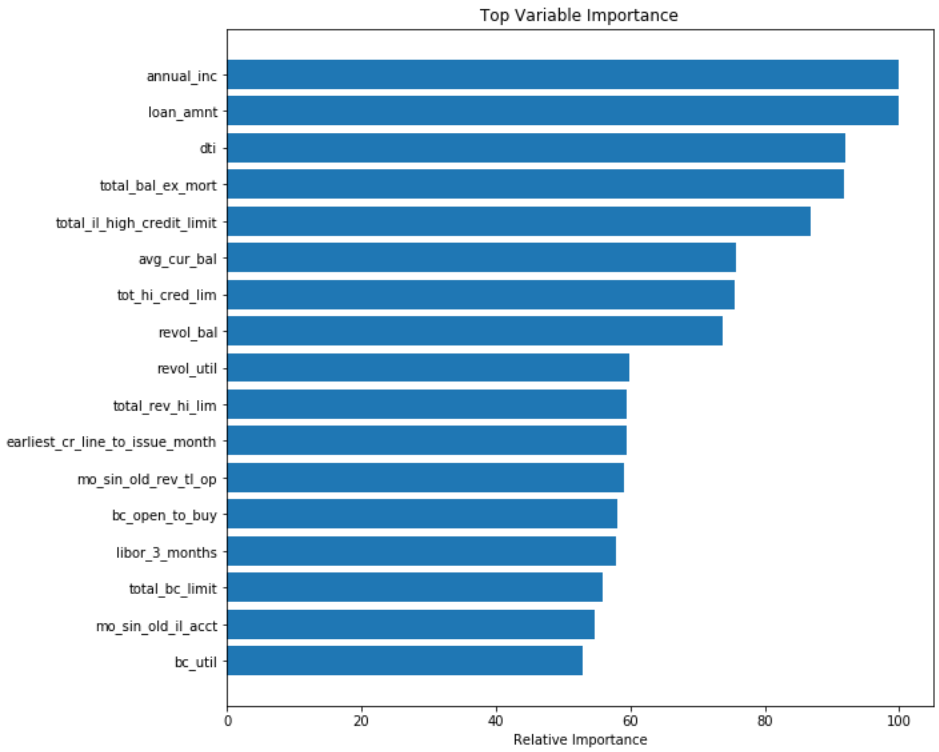

Figure 12: Relative importance values of top performing features as identified by the XGBoost model

Wide and Deep Neural Network

Due to the complexity of setting up and training the neural network model, no grid search was conducted to maximize the AUC value. Instead, the model was setup as suggested by the documentation [15] and was trained for 25 epochs. For each epoch, the entire validation data set was used to evaluate the model performance.

[Figure 13] and [Figure 14] showed the training progress of validation accuracy and AUC as steps increased. The accuracy increased steadily from 0.64 but eventually plateaued at around 0.654. Similarly, AUC had similar increase as steps increased, but the plateau eventually resulted in a small drop of AUC toward the end of steps.

Unsurprisingly, the neural network model scored the highest in accuracy (65.5 %) and Kappa (0.235). Similar to gradient boosted tree, it achieved high accuracy via the balanced classification between default and fully paid loans; in fact, it did a even better job by discerning 67 % of default loans while maintaining a fully paid loan recall score of 0.65 - the neural network model was able to discern both type of loans with similar accuracy instead of over-predicting one of them. Furthermore, the model rejected the least amount of loans (fraction approval rate of 0.584) while maintaining the highest return of investment of -0.040 - a great bal-

ance that minimized the impact to LendingClub's borrowers without compromising profit. [Table 1]

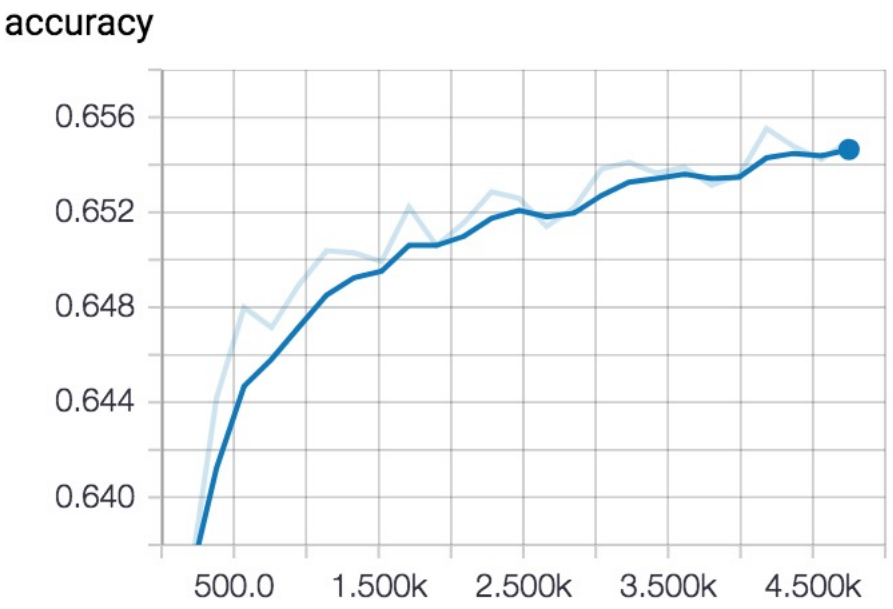

Figure 13: Improvement of wide-n-deep model's validation accuracy over 25 epochs

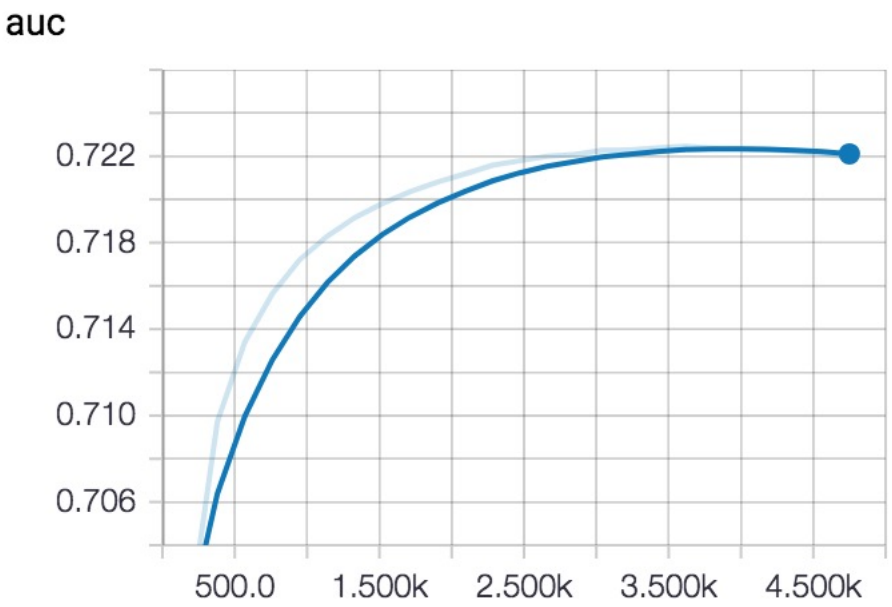

Figure 14: Improvement of wide-n-deep model's validation AUC over 25 epochs

Models	Accuracy	Default Recall	Fully Paid Recall	AUC	Kappa	fraction approved	ROI
Baseline	79.0 %	0.0	1.0	0.500	0	1.0	-0.117
Logistic Regression	63.7 %	0.66	0.63	0.700	0.209	0.571	-0.054
SVM (linear kernel)	63.0 %	0.67	0.62	0.701	0.204	0.561	-0.055
SVM (RBF kernel)	63.3 %	0.71	0.61	0.721	0.223	0.548	-0.043
Gradient Boosted Trees	64.3 %	0.68	0.63	0.722	0.226	0.567	-0.046
Neural Network	65.5 %	0.67	0.65	0.722	0.235	0.584	-0.040

Table 1: Summary performance of all models on validation data set

Test Data Evaluation

Based on the summary metrics for all models listed in [Table 1], the wide-n-deep neural network provided the best performance on validation data set: it achieved the highest accuracy while maintaining similar AUC score compared to other trained models; it provided the highest return of investment (lowest investment loss) without overly rejecting qualifying loans. More importantly, the model was unoptimized with rooms for improvement in terms of feature selections and hidden nodes/layers optimization. Even though the model did not provide any interpretation or inference that could signify whether a loan will have high probability of default, the end goal is simply maximizing investors' profit via the approval of good loans and rejecting bad ones - therefore inference is not necessary important in this case.

The final step of validating the neural network model's performance was through the use of test data set, which was completely hidden from the model training except for the final evaluation. [Table 2] summarized all the metrics of the final neural network model evaluated with the test data, and it's apparent that the performance was very similar to those from the validation data set. It suggested the model generalized very well and did not overfit only to the training data. In summary, the final model was able to predict the loan status of test data correctly 65.3 % of the time. It approved 58.5 % of the loan, while earning a return of investment of -0.0038 - almost broke even if all of these investment were made by a single entity.

Models	Accuracy	Default Recall	Fully Paid Recall	AUC	Kappa	fraction approved	ROI
Neural Network (test set)	65.3 %	0.66	0.65	0.719	0.228	0.585	-0.038

Table 2: Summary performance of the wide-n-deep model on test data set

Simplified RAPPOR

A simplified version of RAPPOR had been implemented, where the step of permanent randomized response was used to mask individual's true response [16]. Due to time constraints, only categorical values had been randomized. [Figure 15] showed the feature distribution of Home Ownership, Verification Status, Loan Purpose, and Employment Length before and after permanent randomized response ($f = 0.5$). It was immediately apparent that the most frequent outcomes in each category were tend to be underrepresented (employed for 10 years or more under Employment Length, for example) while the least frequent outcomes in each category became overrepresented (the rare loan purposes, for example). The randomized response of categorical features from the training data set were trained in a logistic regression along with other non-randomized features, in the same manner as the final logistic regression model shown in previous sections. The original logistic regression model was evaluated with the unmodified training data set, while the new logistic regression model trained with partial randomized response data was evaluated with the same randomized response data. The resulting predictions were compared against each other, and 93.4 % of the training labels were in agreement with each other. Afterwards, the same predictions were evaluated for accuracy using the unmodified training labels. The original model scored an accuracy 63.8 %, while the model trained and evaluated with partial randomized response data scored an accuracy of 63.4 %. The result was unsurprising, considering only 4 out of 46 features had been randomized - the prediction model shouldn't have been affected by a lot. Nevertheless, this shows promises in training an effective prediction model using only randomized data in order to protect consumers' privacy.

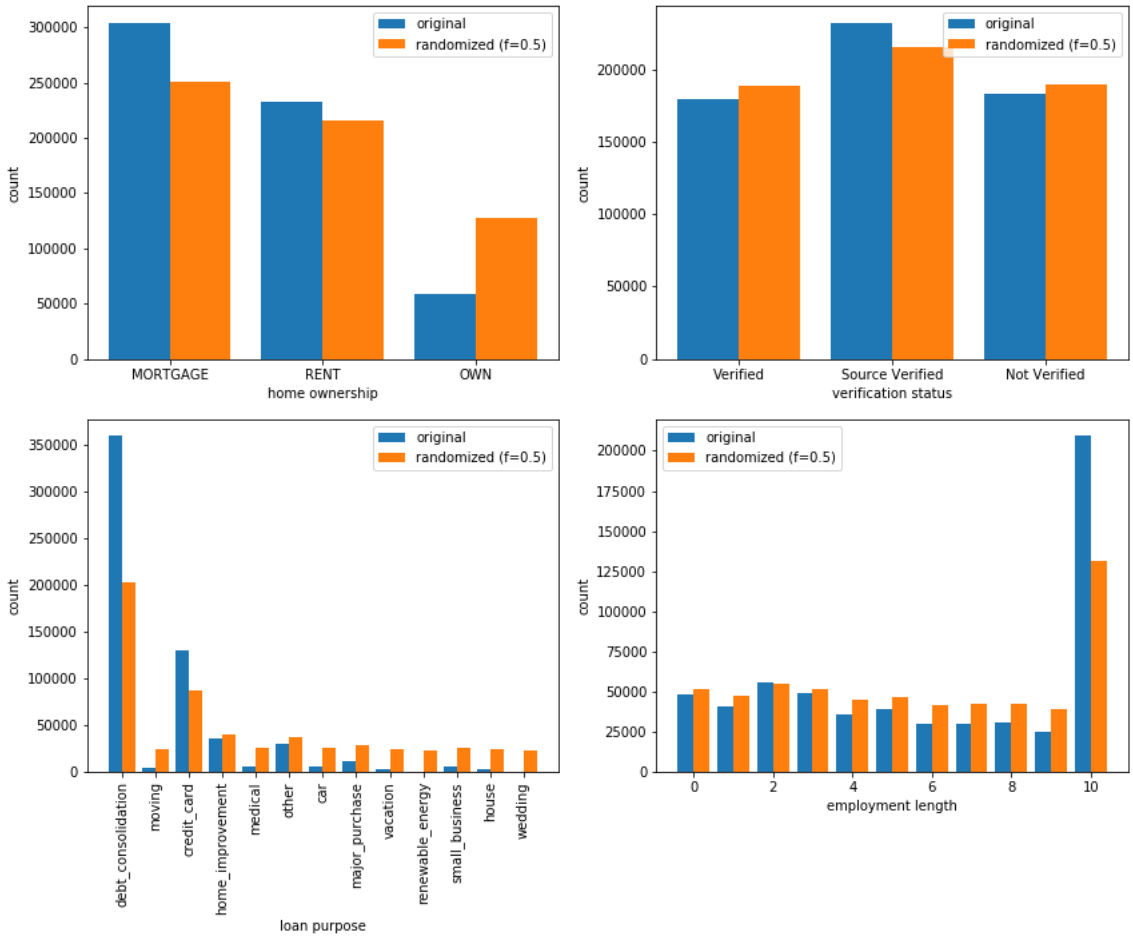

Figure 15: Distribution of Employment Length (by year) before and after permanent randomized response

5 Discussion

This paper has demonstrated the use of machine learning methods to train a prediction model that could discern default loans from other good standing loans with up to 65 % in accuracy. This was achieved using information available before a loan had been approved; this included, but not limited to: borrowers' credit history, unemployment rate, reference short-term interest rate, and more. Nevertheless, the accuracy was nowhere close to the baseline accuracy at up to 79 %. It's of course reasonable to ask: what's the point if blind guessing could do better than a machine learning trained prediction model? First of all, some of the models showcased in this paper provided great interpretation as to why a particular borrower might default on his or her loan. More importantly, inferences could be made on future loans with clear reasonings. From the logistic regression model, one could infer that if a borrower has a high debt-to-income ratio and plan to borrow a high value loan, the loan will very likely to go default. Though the reasoning is very intuitive, it's supported by concrete number and it's something a baseline model and random guessing could not provide. Secondly, it's unreasonable to expect real world loan data to be imbalanced in a way where fully paid loans are always the majority and default loans the minority - if this is the reality, most of the people including the author and readers could be billionaire by following the investment strategy of always buying the majority class of certain stocks. In reality, the data were provided by LendingClub **After** the company had already gone through and rejected many loans that did not meet their underwriters' requirement. It's very possible the unfiltered loans request seen originally by LendingClub could be much more balanced, where the baseline model is no longer be useful. Therefore, models showcased in this paper could be very useful if exposed to unfiltered loans request. Finally, Table 1 had shown that all of the models were able to maximize return of investment, outperforming the current strategy used by LendingClub. Granted, this was at a cost of rejecting close to half the existing loans, it's still a valid strategy to minimize loss if some investors demand a low-risk investment plan.

It's clear that despite the highest accuracy reached by wide-n-deep neural network model, the performance improvement over other machine learning methods was quite minimal. This suggested two possibilities. First, the is the limit with data given to the machine learning algorithms, and higher accuracy could only be reached if more data were to be available - collecting or engineering from scratch. Second, this is due to the fact that the neural network had not been optimized. The second point was certainly true, as an alternative wide-n-deep model that reached 68 % in accuracy was actually trained at one point, but was lost due to computer issues. Given enough time, it's entirely possible to optimize the neural network to the point where there could be a huge margin between it and other conventional machine learning method. Nevertheless, gradient boosted tree struck a great balance between good performance and interpretability - it certainly could be the model of choice if neural network isn't an option.

Finally, a heavily simplified version of RAPPOR was showcased to generate new randomized features and use in machine learning model training. In fact, the similar performance in

terms of train data accuracy between the two models highlighted the possibilities of future prediction models trained using completely anonymized data. This could provide consumers the existing recommendations they already enjoy, without the chance of putting their own privacy at risk.

All things considered, this paper showcased the use of sensitive consumers data (credit report) and various machine learning method to predict whether certain consumers are more prone to default. Certain models provided limited performance but high interpretability, while other powerful methods such as gradient boosted tree and neural network were able to provided better performance with limited interpretability. Furthermore, it was also demonstrated that it's possible to build a loan default prediction model with similar performance using partially randomized user data. This provided possibilities of using a fully randomized response data to build a similar performance loan default prediction model in future work.

References

[1] Kaggle - All Lending Club loan data

[2] LendingClub - Lending Club Statistics

[3] Federal Reserve Bank of St. Louis - 12-Month London Interbank Offered Rate (LIBOR), based on U.S. Dollar

[4] Investopedia - What is 'LIBOR'

[5] Bureau of Labor Statistics - Unemployment Rate

[6] Federal Reserve Bank of St. Louis - S&P 500

[7] Wikipedia - S&P 500 Index

[8] Seaborn - Plotting a diagonal correlation matrix

[9] Stack Exchange - Cohen's kappa in plain English

[10] scikit-learn: SVM Classification

[11] XGBoost - Introduction to Boosted Trees

[12] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, Rohan Anil, Zakaria Haque, Lichan Hong, Vihan Jain, Xiaobing Liu, and Hemal Shah. 2016. Wide & Deep Learning for Recommender Systems. CoRR abs/1606.07792 (2016). http://arxiv.org/abs/1606.07792

[13] Stack Overflow - Produce balanced mini batch with Dataset API

[14] Complete Guide to Parameter Tuning in XGBoost

[15] TensorFlow Wide & Deep Learning Tutorial

[16] Ulfar Erlingsson, Aleksandra Korolova, and Vasyl Pihur. RAPPOR: randomized aggregatable privacy-preserving ordinal response. ACM Symposium on Computer and Communications Security (CCS), 2014. https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/42852.pdf